IN-62-CR
79398
preprint P-7

A RECONFIGURABLE AND FAULT TOLERANT HYPERCUBE ARCHITECTURE[*]

Mary Bucknell

Prithviraj Banerjee

Computer Systems Group
Coordinated Science Laboratory
University of Illinois
1101 W. Springfield Av.
Urbana, IL-61801
(217) 333-6564

(NASA-CR-181044) A RECONFIGURABLE AND FAULT TOLERANT HYPERCUBE ARCHITECTURE (Illinois Univ.) 7 p Avail: NTIS

N87-70462

Unclas
00/62 0079398

[*] This research was supported in part by the National Aeronautics and Space Administration under Contract NASA NAG 1-613.

1. Introduction

The use of multiple concurrent processors that work on the same problem is necessary to obtain large increases in computing speed. The amount of increase depends on the problem, the processors characteristics, and the way the processors are interconnected. Many interconnection networks have been developed in the past, ranging from ring connections to full point-to-point connections. The hypercube network, also called the binary k-cube, has been found to be quite useful for a wide class of problem such as the Discete Fourier Transform and sorting. The hypercube structure consists of an array of $N=2^k$ processors, with each processor connected to its k nearest neighbors. If the processors are viewed as corners of a cube in k-dimensional space, the node-connections or links are the edges of a cube. An advantage of this architecture is that it is a homogeneous, modular architecture allowing the possibility of open-ended expansion. Recently, there has been a lot of interest in actually constructing such cube-connected systems.

The Cosmic Cube [1] system was developed at Caltech and is currently operational. It is a system which uses 64 small computers in a binary 6-cube network. The Mark II [2] is an improvement on the above system with 128 node modules with each module having its own connection to the outside world through a host. Inter-module communication is accomplished through ribbon cables. Both of these systems exhibited good cost/performance ratios. The Mark III computer [3] is being designed to allow up to 1024 processors to be configured as a system. Each node processor operates at a sustained rate of 2 MIPS and over 2 MFLOPS. Recently, Intel has announced the *iPSC* family of concurrent computing systems consisting of 32, 64, or 128 processing nodes [4].

A problem with designing complex systems consisting of such a large number of processors is that the probability of any one or more processors failing is quite large. It is desirable to build some fault tolerance into such highly concurrent systems. Fault tolerant network architectures are therefore emerging as an important area of study [5, 6, 7]. One area that has not been addressed in the design of hypercube architectures is the fault tolerance of such systems. In this paper, we propose a reconfigurable and fault tolerant hypercube architecture.

2. A Reconfigurable Hypercube Architecture

In any reconfigurable connection network it is important that the degree of the nodes does not increase exponentially. The number of redundant processors should also be minimal. The scheme that will be presented requires 2^{k-2} additional processors (for a k-cube) and each node in the system will have degree $k+1$.

As an illustrative example, Fig. 1 shows the reconfigurable 4-cube connection scheme. Processor nodes numbered 0 - 15 together with the links denoted by continuous edges denote the normal 4-cube connection network; processors 16-19 and the dotted edges constitute the redundant processors and links. The original 4-cube network is divided into two groups of eight, namely nodes 0 - 7 (group 1) and nodes 8 - 15 (group 2). Two redundant processors are associated with each group. Processors 16 and 17 are associated with the first group and processors 18 and 19 are associated with the second group. Now consider the binary representations of the processor addresses. The nodes in group 1 with an odd number of ones (odd parity) will have redundant links to auxiliary processor 16 which also has odd parity. Therefore nodes 1, 2, 4, and 7 have redundant links to processor 16 (see Fig. 1). The other nodes in group 1 which have even parity will have redundant links to processor 17. Similarly in group 2, nodes with even parity addresses will have redundant links to processor 18 (which is even parity), and nodes with odd parity addresses will have redundant links to processor 19 (see Fig. 1). In addition to these redundant links there is a redundant link between auxiliary processors 16 and 18, and a redundant link between auxiliary processors 17 and 19.

This configuration can tolerate single node failures. For instance in the example given above, let us assume node 4 has failed. The auxiliary processor associated with group 1 which does not have a link to node 4 is brought in to replace node 4, namely processor 17. The links from processor 17 to nodes 1, 5, and 6 will be activated. In addition the processor in group 2 which has a link to node 4 will be replaced by processor 19. As can be seen from Fig. 1, node 12 will be replaced by processor 19. Therefore the links from processor 19 to nodes 8, 13, and 14 are activated. To complete the reconfiguration the link between processors 17 and 19 would be activated. A similar method can be used to replace any processor which fails. It can be seen from this example that it is necessary to bring in two auxiliary processors and to activate 7 redundant links in order to replace a single faulty node.

3. General Reconfiguration Algorithm

In the paper we will generalize the configuration for any binary k-cube (for $k > 3$). Basically, we start with the binary k-cube connection as discussed earlier. The number of address bits needed to address each processor will now be $k+1$. Partition the nodes into groups of eight (i.e. 0 - 7, 8 - 15, etc.). Each group will have two auxiliary processors associated with it. The addresses of the auxiliary processors will start with 2^k, $2^k + 1$, etc., until all processors are numbered. 2^k and $2^k + 1$ are associated with the first group and so on. Redundant links within groups are as follows:

(1) Nodes with even parity addresses should have a redundant link to the associated processor with even parity address.

(2) Nodes with odd parity addresses should have a redundant link to the associated processor with odd parity address.

In addition, the auxiliary processors have redundant links to each other. Each auxiliary processor with an even address (LSB = 0) has a redundant link to all other auxiliary processors with even address. In a similar way, each auxiliary processor with an odd address (LSB = 1) has a redundant link to all other auxiliary processors with an odd address. It can be seen that every node in the system has degree $k+1$. The auxiliary processors should have four redundant links to their associated group and $k-3$ redundant links to other auxiliary processors. The original nodes should have one redundant link to one of the associated processors within its group.

Formally, a general reconfiguration algorithm for a single node failure is given below. Let us assume that the processor f fails. (where $0 \leqslant f \leqslant 2^k - 1$). The system will be reconfigured as follows:

(1) If f has even parity, the auxiliary processor with odd parity will replace f. If f has odd parity the even parity auxiliary processor will replace f. Call this auxiliary processor L.

(2) The links between L and the processors within the group which were neighbors of k are activated (three links should be activated).

(3) All links from L to other auxiliary processors should be activated. These auxiliary processors will replace nodes whose addresses equal $k \pm 8$.

(4) Repeat step 2. 2^{k-3} - 1 times for auxiliary processors whose links to L have been activated. (let L = each auxiliary processor mentioned in step 3.)

It can be seen that with a single node failure, 2^{k-3} processors will be replaced by auxiliary processors.

4. Reliability Analysis

In the paper we will show the improvement in the reliability of the system topology using our redundant scheme assuming some simple probabilistic models. For the original system to be operational, all the nodes and links have to be operational. In the redundant system, a fault in a single processor Reliability expressions for the nonredundant and redundant systems will be derived. It will be shown that the redundant system is several times more reliable than the nonredundant system.

5. Conclusion

The number of redundant processors needed for this scheme is 2^{k-2}, and the degree of the nodes is $k+1$. In a binary k-cube the degree of the nodes is a limiting factor. Therefore we have chosen to have the degree of the nodes in the redundant network to just one more than that of the nonredundant network.

One possible way to organize a higher order system (one with $k > 3$) is to build chips with a reconfigurable 3-cube structure. Although a layout of this would not be very compact it would be acceptable. The cubes could be connected together through a multi-stage network to form higher order k-cubes. This is easily done since the reconfiguration scheme given in this paper is organized in groups of 3-cubes. Further work is being done to see if it is feasible to organize a k-cube with 2^{k-3} binary 3-cube chips.

REFERENCES

[1] C. L. Seitz, "The Cosmic Cube," in *Comm. of the ACM*. pp. 22-33, Jan. 1985.

[2] J. Tuazon, J. Peterson, M. Pniel, and D. Leberman, "Caltech/JPL Mark II Hybercube Concurrent Processor," *Proc. 1985 Parallel Processing Conference*, pp. 666-673, Aug. 1985.

[3] J. C. Peterson, J. Tuazon, D. Lieberman, and M. Pniel, "The Mark III Hypercube-Ensemble Concurrent Computer," *Proc. 1985 Parallel Processing Conference*, pp. 71-73, Aug. 1985.

[4] Intel Scientific Computers, "iPSC: The First Family of Concurrent Supercomputers," 1985, product announcement.

[5] I. Koren, "A Reconfigurable and Fault-Tolerant VLSI Multiprocessor Array," in *Proc. 8th Int. Symp. on Computer Architecture*. Minneapolis, Minnesota, pp. 425-442, May 1981.

[6] C. S. Raghavendra, A. Avizienis, and M. Ercegovac, "Fault-Tolerance in Binary Tree Architectures," in *Proc. 13th Int. Symp. on Fault-Tolerant Computing*. pp. 360-364, Jun. 1983.

[7] D. K. Pradhan, "Fault-Tolerant Multiprocessor Link and Bus Network Architectures," in *IEEE Trans. Computers*. pp. 33-45, Jan. 1985.

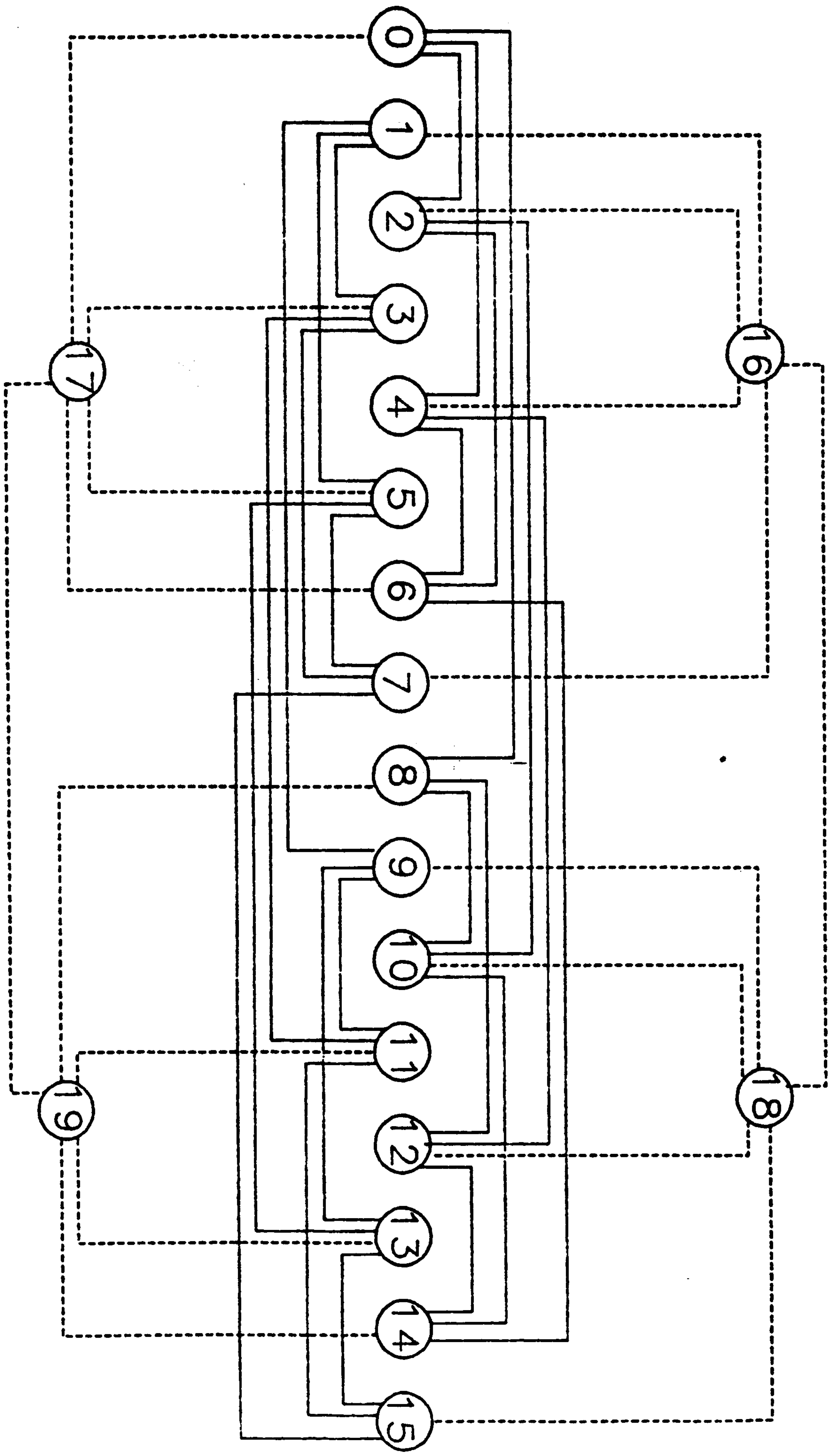

Fig. 1. *Reconfigurable 4-cube connection.*